MODELING EMOTIONAL CHARACTERS IN VIRTUAL ENVIRONMENTS

INFERENCING THE LOOK AND ANIMATION OF A CHARACTER BASED ON AESTHETIC DESCRIPTORS IN REAL TIME

Supervisors: Binh Pham,Ruth Christie
As part of the IT30 Information Technology (Honours) course at
Queensland University of Technology

Table Of Contents

Illustration Index

1 Overview

1.1 Statement Of Research Problem

The problem being proposed for research involves the current difficulties involved in non−technically orientated users making use of complex computer software. This problem is very apparent in the area of computer graphics as an artist usually has to have a detailed knowledge of how the animation system works, and have knowledge on what different effects are available and what they do. This is obviously a problem much too large to be solved within the time frame of this project, so the focus in this project will be on the problem animators can have in giving their characters emotions or moods.

1.2 Rational/Background

Research into the area of choreography [2] has indicated that it is not only possible to express moods by body posture, but that it can be very effective[3]. Historically, the animator had to either be highly trained in arts and computer graphics or the animations produced would look unnatural and have no "emotion" behind them. This is starting to change in current animation systems where increased computing power has led to animation systems being more visual and easy to use. However, it is not yet possible for people with no technical knowledge to produce realistic animations.

Animating a human is a difficult task as a human being has many degrees of freedom which allows joints to flex and skin to stretch and contract in many millions of positions. What makes it worse is that the audience watching an animation can easily detect if an animation is "not quiet right" even if they cannot say for sure what is wrong.[1] This has led to the development of many animation techniques in order to help get more natural movement and expressions. However most research techniques have been specifically designed to only work on one part of the human body (ie legs, face etc) which means these techniques are of little use to an animator. Recently, attempts have been made to combine these techniques in complete systems to allow animators to use them.

When attempting to model a human it could be said, "Why bother with these techniques? Just motion capture a *real* human doing the motion and map control points onto your animation." While historically this is a very successful technique used in many animations[1], it is a very laborious task and typically the same motion is used again and again. This is very noticeable in computer games where a character always runs/walks/jumps exactly the same way over and over again. This method is also unacceptable if you are modelling a virtual environment, as characters will typically need to display an emotion or motion when the system is running that was not thought of when the system was designed.

Another problem that occurs in animation systems is that the system is only as good as it’s ability to extract from the user what they want to do. This problem is starting to be resolved in more modern systems where it is possible to manipulate visual objects to form the animation. However this does not solve the more abstract problem of trying to apply an emotion or mood to the animation.

1.3 Problems In Simulating Emotion In Characters

In order to achieve the goals of the system, there are several individual areas that need to be researched.

Model Animation – Although most of the technical aspects of animating the models will probably be handled by an animation sub−system, it is important that a basic understanding of how these systems work is achieved so they can be modified to achieve the research goals. This information will also be useful in selecting an appropriate animation system.

Applying Emotions to Models− In order to create a system to apply an emotion to an animation, a knowledge of current methods and performance of these methods is required.

Inference Engines− To adequately create a realistic virtual environment that is to the user’s expectations. It is obvious some sort of inference engine will have to be used. This way abstract data (such as an emotion) can be mapped onto the animation and the virtual environment.

1.4 Research Aims And Objectives

The aim of this project is not to build a complete animation system for an animator to use. As stated previously that would be too much to complete within this project’s time frame. The system to be developed will focus on the user being able to specify an emotion state or mood of a virtual environment and the characters that interact in the environment. The system will then manipulate the model’s stance and motion to incorporate these moods/emotions. It should be noted that this is not a system to generate animations for a given model, but to modify a model’s animations to incorporate specified emotions. Body expression was chosen as facial expression has been done [19] and carried out successfully in many large budget projects.

The interface for such a system will obviously have to be very easy to use so an inference engine built into the virtual environment will be needed. This inference engine will help take simple user data and then incorporate it into the emotions of the system.

This project will also aim to use as few computational cycles as possible as to allow the developed techniques to be used as part of a larger system. (ie a game)

This research will be done in conjunction with another system being developed to manipulate the environment variables based on these emotions/moods.

1.5 Significance Of Study

The significance of this study is that applying emotions in real time to a character has not been done to any great extent previously. This could therefore lead to better animation tool design and more life like characters that interact in computer games. However this will depend on how much computational power is eventually to perform the model deformations.

1.6 Limitations Of Study

By far the greatest limitation of this study is time. It is this factor that limits the number of emotions that can be incorporated into the system. This is due to the fact that an emotional pose takes a significant amount of experimentation and in the six months allocated for implementation, only a handful of emotions will able to be modeled. Also if more resources were available (such as access to real human movement data and a wide variety of models), more input data for the emotion models could be acquired to achieve greater realism.

2 Literature Review

The aim of this literature review is to overview current techniques for animating models, applying emotions to models and inference engines. This is to reveal what has been done and problems that have been encountered with these techniques.

This review is also to show that combining these techniques in a virtual environment scenario has not been done previously to any great extent.

Since many of the discussed techniques are extensions of, or require knowledge of, other techniques the organisation of the literature review in each section is from the general techniques to more specific techniques. Each section addresses one of the four sub−problems mentioned in the Problem Definition.

2.1 Common Model Animation Techniques

Research into animation modelling techniques has shown that there has been developed many varying and unique attempts at the problem. While no technique conclusively solves all of the problems that can arise, a combining of some of these techniques can usually produce the desired result in terms of speed, efficiency and accuracy of the produced animation system.

All the discussed techniques assume you have a basic knowledge of computer graphics especially in the areas of vertices and 3D meshes.

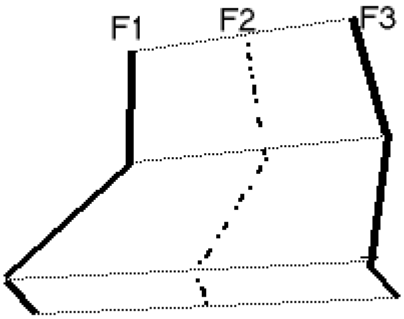


Illustration 1

Frame 2 is interpolated between
Frame 1 and Frame 3

2.1.1 Simple Shape Interpolation

Simple Shape Interpolation (sometimes referred to as shape blending or multi target morphing) is by far the easiest and most implemented animation modelling technique there is.[4, 5] This technique works by the user simply specifying the position and orientation of the surface control vertices at certain key frame positions in the animation. To calculate the frames between the given frames, the technique simply interpolates between corresponding vertices.(see Illustration 1) Linear interpolation is commonly used in the case where speed is required but using other interpolation techniques that involve curves such as splines can produce a more natural result.

As mentioned, the greatest advantage of this technique is it’s ease of use and implementation as well as it’s ability to generate the animation rapidly. This system is popular in animation packages as it gives the animator the ability to ’sculpt’ important frames and have control over the animation. However, there are many occasions when the interpolated frames are not acceptable (ie. such as a hand passing through a wall) and the animator has to manually define some more in−between frames. In a worst case scenario the animator may have to define *every* frame for the animation sequence in order for it to look acceptable. This defeats the purpose of animation modelling system.

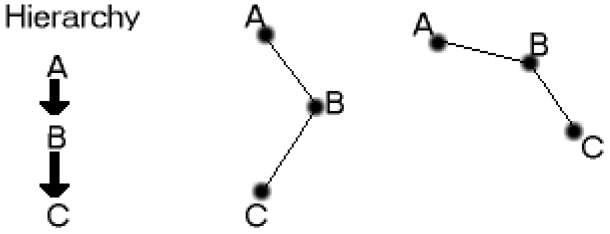


Illustration 2

A rotation in the root joint A causes all descendant joints to rotate relative to A

2.1.2 Skeletal Deformation

In order to understand skeletal deformation type animations, an understanding of the skeletal modelling is required. While this is discussed in many graphics books and references [5, 6], a simple explanation follows.

The basic principle of skeletal modelling is a series of rigid objects(bones) connected by joints that can have multiple or limited degrees of freedom to move or rotate. The segments of the skeleton can then be arranged in a hierarchical structure where rotations on the root node are applied to all the segments further down in the tree. (ie a rotation in the leg node will rotate the foot and all the bone segments) (see Illustration 2)

The animation types discussed deal with kinematics as other types of skeletal model animation techniques (such as Dynamics) were not considered relevant to this project.

Forward Kinematics

Forward Kinematics [6, 7] uses the skeletal model in combination with the interpolation techniques described previously. Instead of positioning the vertices for each key frame, the animator adjusts the joint rotations and movements in the skeleton. Interpolation techniques are then applied to the joint angle rotations and positions to achieve the intermediate frames.

This simple technique is well established and inherits all the speed advantages of shape interpolation but unfortunately, it inherits many of it’s problems as well. This can lead to frustration of the animator, as they have to manipulate many joint rotations on many frames to achieve realistic animation.

Inverse Kinematics

Inverse kinematics[6, 8] differs from forward kinematics in the way an object's location and orientation is determined. Instead of an object within the skeleton being positioned by the rotations and positions of objects higher in the hierarchy, the object can be positioned *manually* and the joint rotations for objects higher in the hierarchy are calculated for the object's desired location. This technique has problems as not all desired locations are possible and there may be several solutions to the placement of higher order objects. Unfortunately this method can be computationally expensive when working on a large skeleton with many degrees of freedom and still exhibits the problems of forward kinetics.

2.1.3 Skeleton Subspace Deformation

This technique has been implemented widely in animation packages but few technical articles exist about it. Skeleton Subspace Deformation[4] (or Skinning) uses the skeletal model techniques previously discussed but uses an outer mesh (or skin) around the skeleton model. Vertices in the mesh are linked to objects in the skeletal structure so that if the skeleton object moves, the surrounding mesh moves with it. This technique has the advantage of the animator being able to create animation without having to worry about proportions of the animated figure changing during the animation. Also since it is only the underlying skeletal model being animated, the computations can be calculated quickly and be stored in a compact form. For these reasons, this technique can be seen in modern computer games.(ie Quake, Tomb Raider etc.) However, the animator not only has to take into consideration the problems with skeletal animation but also the problems that can occur with the mesh at joints that are rotated at large degrees. At such joints the mesh can stretch and deform in strange ways that makes the model look greatly different from the original state.

2.1.4 Generic Algorithms/Layering

There exist many small generic techniques that can be incorporated into an animation system in order to aid the animator and add more realism. Gravity and wind effects are some of the more common techniques and can use real or ad−hoc physics implementations .A typical animation system would usually combine or *layer* a few of the above techniques and generic algorithms in order to reduce the disadvantages that can arise in some techniques. This can help reduce the effect of characters that seem to “float ” along the ground surface.

2.1.5 Conclusion

In this section many animation techniques for mesh animation and manipulation have been discussed however there exist many more techniques. (ie Muscle Animation) The selected techniques were chosen as they are typically fast and well established and therefore well suited to a virtual environment system.

However in this project, skeletal subspace deformation will be the most useful as it provides a direct way of modifying a models pose.

2.2 Applying Emotions To Models

Being able to model an animation is one thing, being able to apply an emotion to that model brings about a completely different set of problems.

One of the main problems is to do with what people consider emotion to be and what it should look like. This is not just a problem for computer animators, as it has been encountered many times over by 2D cell animators for some time now. However solutions that have been found cannot usually be defined by a finite algorithm. What usually occurs is the animator bases the emotion from a real life drawing or memory and adjusts it until the result looks adequate. This technique is so effective that cubes and teapots can be made to express emotion.

The problem is compounded in the 3D computer graphics system to be developed as not only is there an extra dimension to deal with, but we have to apply the emotions to an existing 3D model. To simplify matters this section will only deal with a biped humanoid model.

2.2.1 Acquiring Emotion Data

Most research into emotion modelling capture example data of the required emotion from a real life actor. This data is then used as a basis for the generation of their emotion model by measuring differences between a neutral stance and the emotive state. This process is laborious and generally produces the most realistic results but unfortunately this project will probably not have access to such resources. The only alternative is trial and error of varying input data and using external observers to gauge the results.

2.2.2 Blending/Combining

This method simply takes two animations or a static model and an animation which are of the same object. The model then interpolates between them to produce an animation sequence that combines motions and poses of the figure. This technique is by far the simplest but is rarely used by itself due to it's inability to produce acceptable results directly. This method can be applied to both a surface mesh and an underlying skeleton model and does not have to combine each mesh evenly. Due to the unnatural result of this method, it is generally used in conjunction with several model rules[7, 9]. (ie such as feet must always be touching the ground, head cannot turn 180° etc)

This research project will probably involve some form of blending to combine an emotional model with an animation due to the technique's simplicity and speed. However, the influence this technique will have over the resulting model will have to be determined by much experimentation.

2.2.3 Motion Retargeting

Motion Retargeting [10]involves applying a previously captured skeletal animation to a new skeletal model that may be geometrically and topologically different. One recently proposed solution [8] involves mapping the input skeleton onto an intermediate skeleton before applying the motions to the final skeleton. Inverse Kinematics are then used in order that the end points in the final skeleton match up to the end points in the input skeleton. This technique can produce very realistic results but does involve some guess work by the system or interaction from the user to map points to the intermediate skeleton.

Motion Retargeting will probably form the basis of many initial algorithms in this research as in the virtual environment, it is desirable that these emotions work on an arbitrary humanoid form.

2.2.4 Frequency And Amplitude Application

Often frequency and amplitude of a motion can determine the emotion expressed by an object. (ie knocking on a door angrily would involve high velocity and high joint movement of the arm) The ingenious idea that such frequency and amplitude data could be mapped from one motion to another was first proposed in the paper “Emotion from Motion” [1]. This technique works very successfully as emotion in the form of frequency and amplitude can be extracted from a motion such as door knocking, and applied successfully to several other motions such as kicking and drinking. The technique is usually applied to an underlying skeletal model as a method for manipulating joint angles is required.

The algorithms behind this technique can be very complex but a simplified cut−down approximation version could suit this research very well. This would help enable the virtual environment modify non−humanoid models.

2.2.5 Random Noise

One of the main problems in virtual environments such as games is that the characters are always repeating the same motions again and again. This means that even if the animation is very realistic and emotional, it starts to lose it’s realism once it becomes predictable. A simple solution to this problem is to add some random noise to the animation each time it runs. [7] Emotions could also be simulated with this technique as fast random data could indicate a nervous or scared person. This technique can be very successful but care has to be taken that end points of the animation remain near constant so that “grabbing” animations will still work. Applying inverse kinematics could easily solve this problem.

As this is a very simple and efficient technique, it will certainly be included into the virtual environment in combination with some of the above techniques.

2.2.6 Conclusion

The techniques mentioned above seem to be applicable in expressing emotions in this research project in one form or another, but they do not address the problem of how different people perceive emotions. Perhaps a database of different settings for each type of emotion may be appropriate in a large−scale system, but unfortunately that is beyond the scope of this project.

Combining the motion retargeting, frequency and amplitude and random noise techniques into the project's system will be feasible as each technique is simple, fast and very effective. The proportions of this combining will have to be discovered later by experimentation.

Ideally emotion data could be motion captured from many different sources and standardised into input data, but the amount of effort required for this makes it unfeasible in such a short project. A trial and error approximation method will have to be acceptable for this project.

If CPU power was unlimited much more detail could be put into the techniques for expressing emotion. This is especially applicable with techniques that use inverse kinematics as this operation can be CPU intensive. However, focusing on humanoid emotion animation does compensate significantly in the ability to approximate.

2.3 Inference Engine

Having successful modelling techniques only solves half the problem as a system using these techniques is only as good as it's ability to extract from the animator what they want to do. Previous animation systems can produce very realistic animations but request from the user a database of input data (typically involving data such as muscle groups, strength, torque, mass, velocity etc). This obviously makes the system unusable to the average user. What is needed is an inference engine to infer from crude user input, the data needed by the animation system. This would enable the construction of a user interface that meets the standard types of animations that a user would typically create.

While complex types of inference engines are available, (ie Neural Network[11]) these systems would probably be inappropriate for such a short project as significant levels of experience would be needed to design, create and train such a engine. Therefore this section will explain some of the simpler ways of inferring data and what type of data to infer for emotions.

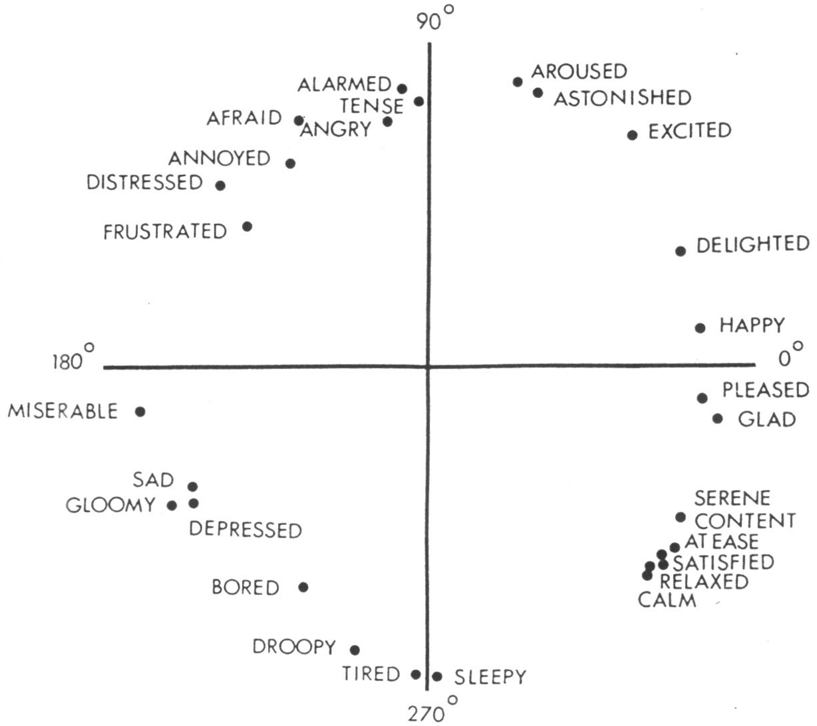


Illustration 3

A comparison of how different emotions relate to each other.

(from figure 2 in "A Circumplex Model of Affect" [12])

2.3.1 Emotional Model

Finding out the relationships between emotions is crucial where emotional factors will be interacting. Fortunately, extensive psychological studies have been carried out in this area [12] and have produced scaling diagrams (Illustration 3) where emotions can be compared in relation to each other. These diagrams also provide ideas on what types of emotions would be most appropriate to be modeled in the system.

2.3.2 Direct Mapping

Direct mapping is a simple technique where the user specifies the emotion they require for the model and the data is applied to a model via a direct look−up in a database. The user may also specify some multiplier value in order to scale the applied database data. This technique is easily implemented in systems that support scripting such as demonstrated in Improv[13]. Application of this technique is very fast but a scene with many objects with different emotions may look inconsistent.

This technique will probably be implemented in combination with other methods in order to give the user some direct control over objects in a scene.

2.3.3 Environment Context

The Environment Context technique involves the setting of the mood/emotion of an entire scene and when objects are placed in the scene, they adapt to the surrounding mood. Each character/object placed can have a variable, which determines how much it is affected by the mood in the scene and the other objects within the scene. The object can also have a variable to determine how much it affects other objects within the scene. The emotion of an object can be manually tweaked using direct mapping which will then influence other objects in the scene. This technique ensures a consistent look over the entire scene and is not too computationally expensive.

This technique is the most appropriate for this project as it is fast and allows an animator to key in as much or as little data they feel is required for the scene.

2.3.4 Conclusion

It is obvious that the quality of a virtual environment system can be determined by the usefulness of it’s inference engine. Because a system with the best modelling algorithms is useless if it cannot infer the wishes of the animator.

It is also important to understand how emotions relate to each other. (ie if a system implements emotions that are all too closely related, the resulting system will not be able to generate a wide range of emotion environments).

The two techniques discussed for the inference engine design are very appropriate due to the speed and simplicity of their implementation. However using an environmental context may produce a more appealing result.

2.4 Related Software Systems

In order to prevent “re−inventing the wheel”, knowledge of what current systems relate to this project and what tools are available to aid in implementing it must be investigated.

Although this is a relatively new area of research, there exist systems and tools that are related to some aspects of this project. These vary from inference engine implementation to graphic technique examples and tools.

2.4.1 Improv

Improv is a system for scripting interactive actors in virtual worlds.[13] Improv consists of two sub−systems: an animation engine and a behaviour engine. The animation system uses animation blending to combine actions, which then can be grouped and classified. This enables data about what actions can be performed simultaneously to be stored. The behaviour engine works by operating on plain English scripts and decision rules. These scripting rules can then be built up to construct a complex virtual environment.

Improv relates closely to this research as it contains both an animation system and the crude beginnings of an inference engine. However this project aims to create a more specific system focusing on humanoids and speed of execution.

2.4.2 Swamped!

Swamped! [20] is a virtual world populated with human and computer controlled actors. It is a sophisticated system that has been upgraded with many different features. These include a plush doll character interface and a camera creature, which examine a scene to find the best fitting camera shot.

Swamped! is probably one of the best examples of a virtual environment that exist today and provides much inspiration and information on what a virtual environment should contain.

2.4.3 Catz/Dogz

Trainable pets have been a recent fad, however one such system is still going strong. This is the Dogz/Catz software made by The Learning Company where the objective is to train a dog or a cat. The emotions exhibited by these pets can be quite believable and that is what makes this software so addictive.

The Catz/Dogz software relates to this project as they are able to simulate an emotional model using computer graphics. However this simulation is done in 2D so therefore the same principles cannot be directly applied to this project.

2.4.4 Game Engines/Auran Jet

The simplest way to create a real time virtual environment is to use a game engine. This is because they are specifically designed to model a virtual world in real time. There are literally hundreds of game engines to choose from (ie Genesis 3D,Quake etc) but this project requires that not only the engine be able to do some or all of the modelling techniques described above, but cost little or nothing. (ie Quake engine costs $500 000)

Auran Jet meets these criteria as it implements simple shape interpolation and forward and inverse kinematics. Auran Jet also costs nothing to use for non−commercial purposes. The models in Auran Jet are also able to be exported from popular 3D modelling software such as 3D Studio.

2.4.5 Conclusion

As shown above, there does not appear to be a lot of software available that directly relates to this research. This is understandable as many of the found literature is less than five years old. (Many less than a year old) However, the basic tools for model animation exist to enable the focus of the project to lie solely on the emotion state of the actors and the environment.

3 Model Development

3.1 Modifier Development

As shown by the above research, there are several ways that models can be manipulated in our goal to achieve an emotional look. These manipulations have been combined into a series of "modifiers" that should achieve the goal of emotional appearance with real time speed. These modifiers aim to achieve this emotional look applied onto an animation without making the model appear distorted. The extent to which a specific modifier is applied, is based on an input parameter that indicates a percentage factor (ie 0% −no modifier applied, 100%− current input is completely overridden by modifier data) Values outside the range of 0−100 are permitted as they can produce some strange (and often humorous) results that may be useful in some games/surreal virtual environments.

A summary of all data required by each modifier is also presented to emphasize what data is needed and what the input source is.

3.1.1 Pose Blend Modifier

The principle behind this modifier is the simple premise of blending pre−generated pose data for the required emotion. This obviously has the disadvantage of having an animator pre−calculate the pose positions for each emotion that is needed to be applied to a model. However, since this project is mainly focusing on human pose, a database of general pose positions for humans can be generated to be mapped onto specific input models.

The resultant position/orientation for each bone is simply calculated by a linear interpolation between the input bone date and the corresponding pose bone data. The interpolation for orientation uses the SLERP (Spherical Linear Interpolation) algorithm for quaternions while the position vectors simply have their numerical x,y,z values linearly interpolated.

An addition per−bone parameter can be included in this modifier to indicate the importance of a particular bone in influencing the emotion. This additional parameter is then combined with the global percentage factor before the interpolation is calculated.

This has the advantage of non−essential pose data not affecting animations that may occur in other parts of the model’s body. This extension can easily be calculated by comparing the emotion pose to a neutral pose and calculating the difference on a per−bone basis. This therefore requires an additional neutral pose or extra data stored in the emotion pose. This extension could also be applied to only specific roll, pitch, yaw movements on the bone.

Data Parameters	**Input source**
Global influence	Real time input
Orientation tolerance limit	Load or compile time variable (for pose comparison)
Position tolerance limit	Load or compile time variable (for pose comparison)
Pose data (per−bone and per−emotion)	Pre−calculated from file
Neutral Pose data (per−bone)	Pre−calculated from file

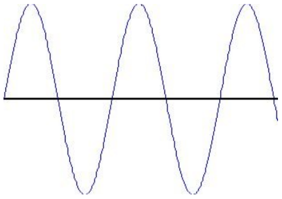


Illustration 4

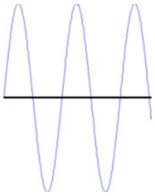


Illustration 5

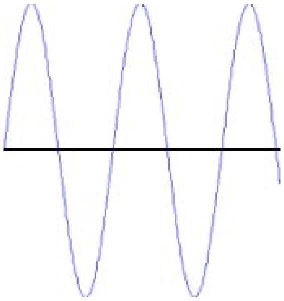


Illustration 6

If Illustration 4 is the path of bone movement, Illustration 5 has increased frequency with Illustration 6 having increased amplitude (straight line is amplitude 0)

3.1.2 Amplitude And Frequency Modifier

As previously discussed, modifying the amplitude and frequency of a model’s animation can drastically alter the perceived emotion. (ie gentle light knocking vs fast hard knocking – same motion, different amplitude and frequency.) As illustrations 4, 5 and 6 show, this effect seems relatively simple. While varying the frequency is a simple matter of altering the animation speed, calculating different amplitudes is not so easy.

The main problem in calculating a new amplitude is finding where amplitude equals zero should be. The obvious answer is to pre−calculate where the neutral pose for this model is and use each per−bone position/orientation as the amplitude zero position. Each input bone is then compared against the neutral pose bone and has its amplitude linearly interpolated by the global influence variable. This method works quite successfully on its own but works in a fashion very similar to the pose modifier and may counteract the pose modifier changes if both modifiers are used at once.

Perhaps the most obvious method for amplitude zero determination involves simply finding the mean of all positions and orientations in the animation. This method has the advantage of being general as it does not require any pre−calculated data for the model and can be written to adjust the average calculations at run time or access the animation data and calculate the means instantly. Adjusting the average at run time has the advantage of being able to handle changes in the animations being played but takes a full animation cycle to stabilise. Therefore a combination of both should be used with this method. (ie calculate initial average and adjust while the animation is playing). However, while this method is sound in theory, applying it to a human model can produce strange results (ie bent knee as neutral position in running)

The next obvious method to calculate the amplitude zero position is to use the previous frame’s bone position/orientation. This would mean that each bone would be compared against its previous position and interpolated accordingly. However, all this algorithm seemed to achieve is a result similar to the previous mean/average method.

While the previous methods do work, the method decided upon for this modifier works on the premise of each bone using one of their positions/orientations in the animation. This method involves comparing each bone’s orientation before the modifier is applied against a pre−determined fixed orientation. If it’s orientation is then closer than the current saved orientation, the new position/orientation is saved and used as the zero amplitude position/orientation. The pre−determined fixed orientation can simply be (0,0,0) in Euler angles (or no movement) as most models are created around this zero orientation (or are close to it). Otherwise, the neutral pose orientation can be used if the model is not created like this. This method has the disadvantage of that the position data is not used in these calculations (but is modified in the same way as the orientation so that distortions do not occur) but this is not a major concern as most skeletal animations involve mostly orientation movements. The main advantages of this method is that it does not assume a neutral position that is out of scope for the animation movement and that it can be applied without any pre−defined data.

Data Parameters	**Input source**
Global influence	Real time input
Frequency Influence	Real Time input
Amplitude Influence	Real Time input
Amplitude Zero data (per−bone)	Calculated during run−time
Neutral Pose data (per−bone) (optional)	Pre−calculated from file

3.1.3 Gravity/Direction Modifier

The basic premise behind this modifier is that many emotions give the appearance that the character is being pulled or pushed in a certain direction. (ie. Depression gives the appearance that gravity is pulling extra hard on a character, while happiness almost makes the character defy gravity’s effects.) However this modifier is not just limited to pulling downwards. For example a pull direction from behind a character gives the appearance that the character is struggling to move against a forward wind.

One way of implementing this modifier is to compare each bone’s position and parent bone’s orientation (as the parent bone’s orientation effects the position of the child bone) against the previous position and orientation. From this a direction vector can be calculated to indicate the movement of the bone from the previous position. The movement direction vector is then projected onto the vector indicating the gravity direction. The result of this yields a number indicating what proportion of the movement was in the gravity’s direction. This proportion number is then combined with the influence for this modifier to scale the orientation and position changes for the bone.

However due to problems in retargeting an orientation at a new position using Auran Jet, this method was unable to be implemented successfully.

A simpler method that was able to be implemented involves a technique similar to the Pose modifier. This works interpolating (using the influence value) the positions/orientations of each bone against saved bone positions/orientations that are close to a specified gravity position.

The saved positions/orientations are generated by comparing each bone’s final position against the gravity position. If the bone is closer than the previously saved data, the bone’s local position and parent orientation is saved. This is repeated for all frames in an animation sequence and once again can take a full animation cycle to stabilise if the initial data is not pre−calculated from the animation sequence.

While this method does work, the results can sometimes be strange and is not recommended to be used with high influence values.

Data Parameters	**Input source**
Global influence	Real time input
Gravity Position	Real time input
Saved Gravity data (per−bone)	Calculated during run−time

3.1.4 Random/Noise Modifier

As discussed previously, animations that are continually repeated on a model will start to loose their emotional appearance. This is because the animation soon becomes predictable to an observer. A solution to this is to add some noise/randomness to each frame of the animation. However, care must be taken to ensure that the animation still looks consistent and smooth.

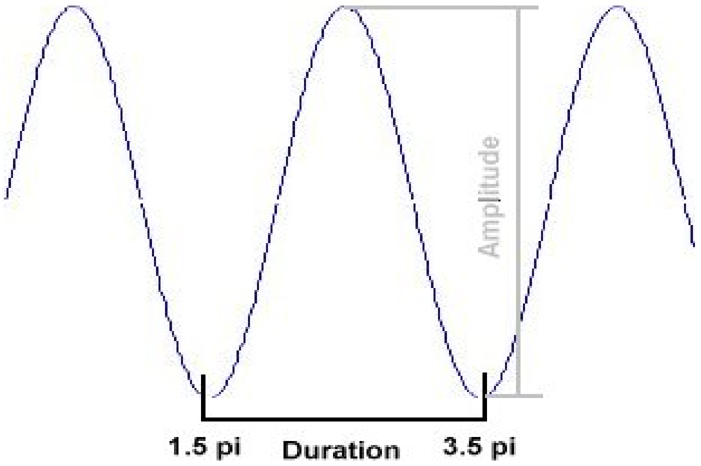


Illustration 7

The Sine wave added to smooth random motion

$$interpolate_{value} = \frac{\left[(\sin([(\frac{current_{bonetime}}{total_{bonetime}}) * 2 + 1.5] * pi) + 1) * current_{influence} \right]}{2}$$

Illustration 8

The method used in this implementation involves randomly adding Sine waves to bones in the animation sequence. As illustration 7 shows, Sine from 1.5pi−3.5pi radians is used to achieve the smooth addition of randomness. Illustration 7 also shows there are two data parameters need to model the Sine wave, the variance/amplitude and the duration. The Sine wave is applied to a chosen bone by creating variance values for each Euler angle (pitch,roll,yaw) and doing a interpolation between the current orientation and the variance values added to the current orientation. Illustration 8 shows the formula that is used to calculate this interpolation value. As shown, this formula makes use of a selected duration(total bone time) of the bone as well as the amount of time that has elapsed since the bone began to be modified.

The variance to use is determined by the user providing a maximum variance value which the program then uses to randomly assigns pitch, roll, yaw numbers between

(−max_variance,max_variance). For the duration, the user provides a minimum time to maximum time and the program makes a selection between these values. Both variance and duration is unique for each bone being modified in the current implementation.

When deciding what bones to modify, the program takes a value which indicates what percentage of bones are to be modified randomly at any one time. Initially the full percentage of bones is randomly allocated and as each bone completes modification, a new bone not currently being modified is randomly selected. A newly selected bone also has new variance and duration values calculated for it.

The above method does perform acceptable modifications however, bone movements are not constrained and the system has no way of deciding what movements are valid for a model. This occasionally leads to bones moving where they should not (ie hand though body). The implementation of this method constrains movement on root bones so the model is not rotated on it’s own axis. Also, this method makes no use of bone position data which may need to be considered in models that make extensive use of position movements.

No further methods were tried due to time constraints and apparent effectiveness of this method. However, further attempts should be made to implement other methods to form a comparison (ie the Ken Perlin method [13])

Data Parameters	**Input source**
Global influence	Real time input
Variance	Real time input
Duration min,max	Real time input
Percent random	Real time input

3.2 Inference Engine

The design of the inference engine was aimed at allowing the user to specify a minimal amount of information to apply the above modifiers to a scene. However, it has also been designed so that if a user wishes to specify more information, they are able to create a more realistic scene.

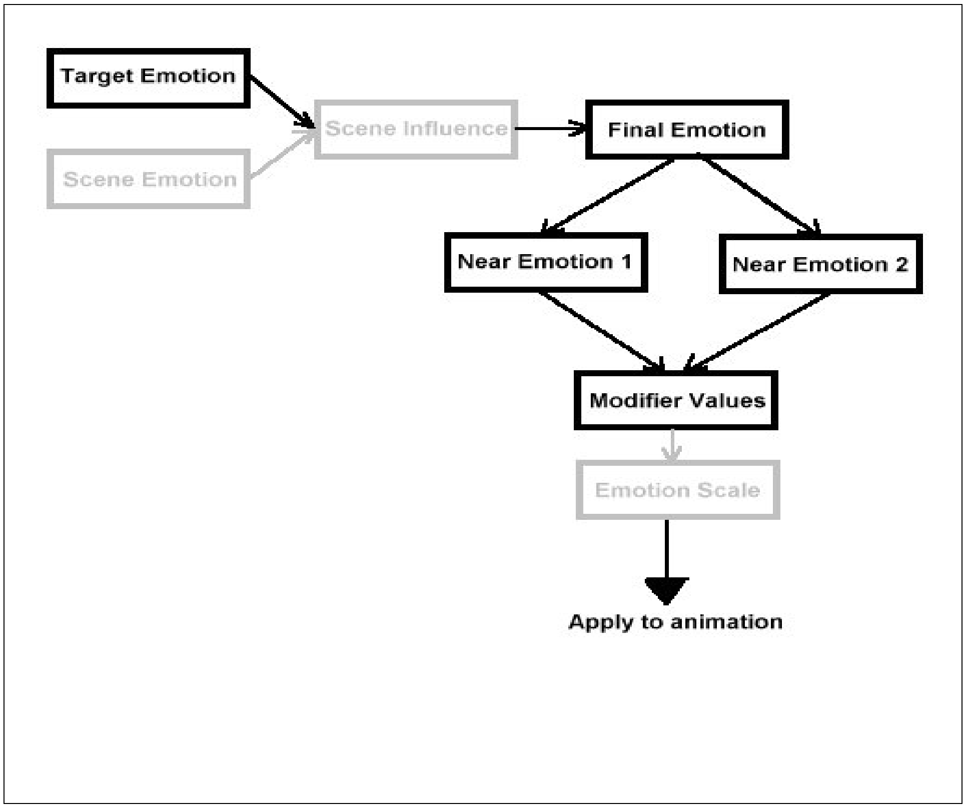


Illustration 9

The inference engine flow diagram.

The inference engine begins by assigning each valid emotion a degree value ($0^0 - 360^0$) in positions similar to that seen previously in illustration 3. Then the inference engine processes the user input as displayed in illustration 9.

The processing starts with converting the emotion the user wants to be displayed on the model to a degree value. This value then can be optionally mixed with an emotion that has been specified to represent the mood of a scene of models. This mixing of emotions also requires a value to indicate what proportions to mix the emotions (ie 40% scene mood, 60% model emotion). The mixing itself is a simple interpolation of emotion values using the proportion as an influence (ie emotion at 0^0 + emotion at 90^0 with 50% influence = emotion at 45^0)

From this result the inference engine then checks what two pre−programmed emotions the value falls between. These pre−programmed emotions have well defined input values for the modifiers. (Due to time constraints, implementation of only four pre−programmed emotions was obtained) The inference engine then interpolates both sets of modifier input values based on how close the pre−programmed emotion is to the target emotion. This modifier data is then added together to produce approximated modifier data for the target emotion.

In the next step an additional value may be used which specifies how much emotions can effect a model. This input simply scales all the modifier data before being sent to modify the animation.

This inference engine does produce acceptable emotion output and even better results may be obtained by pre−programming more emotions than the current four. There is however a minor issue with this model when the scene emotion and target emotion are at almost 180^0 to each other with scene mixing approximately 50%. It would be logical that in such a situation the emotions would cancel each other out but this inference engine does not take this into account. The result is a emotion half way between the two emotions that can suddenly flip 180^0 if the target or scene emotion are changed by 1^0. This situation however is quite rare and a hard wired change to the inference engine to handle this occurrence should resolve it.

4 Implementation Issues

4.1 Module Interfaces /Class Relations

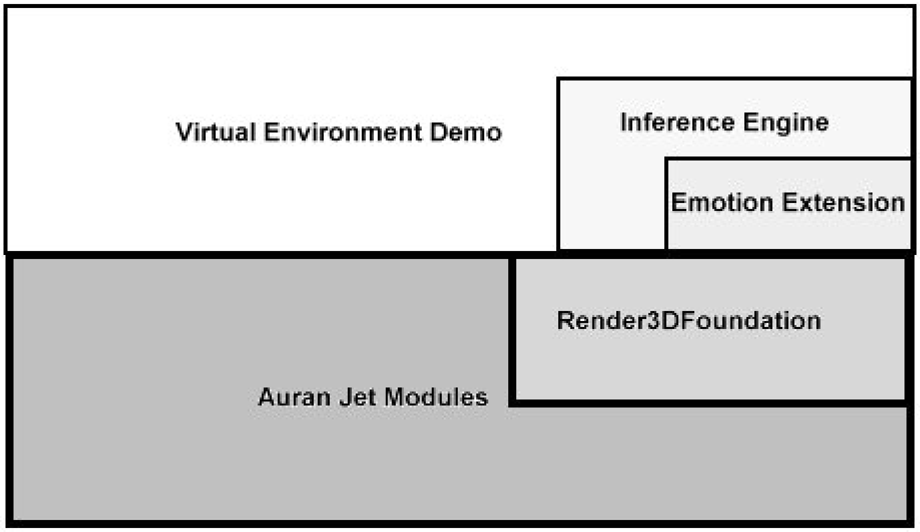


Illustration 10

How the modules fit together on top of Auran Jet

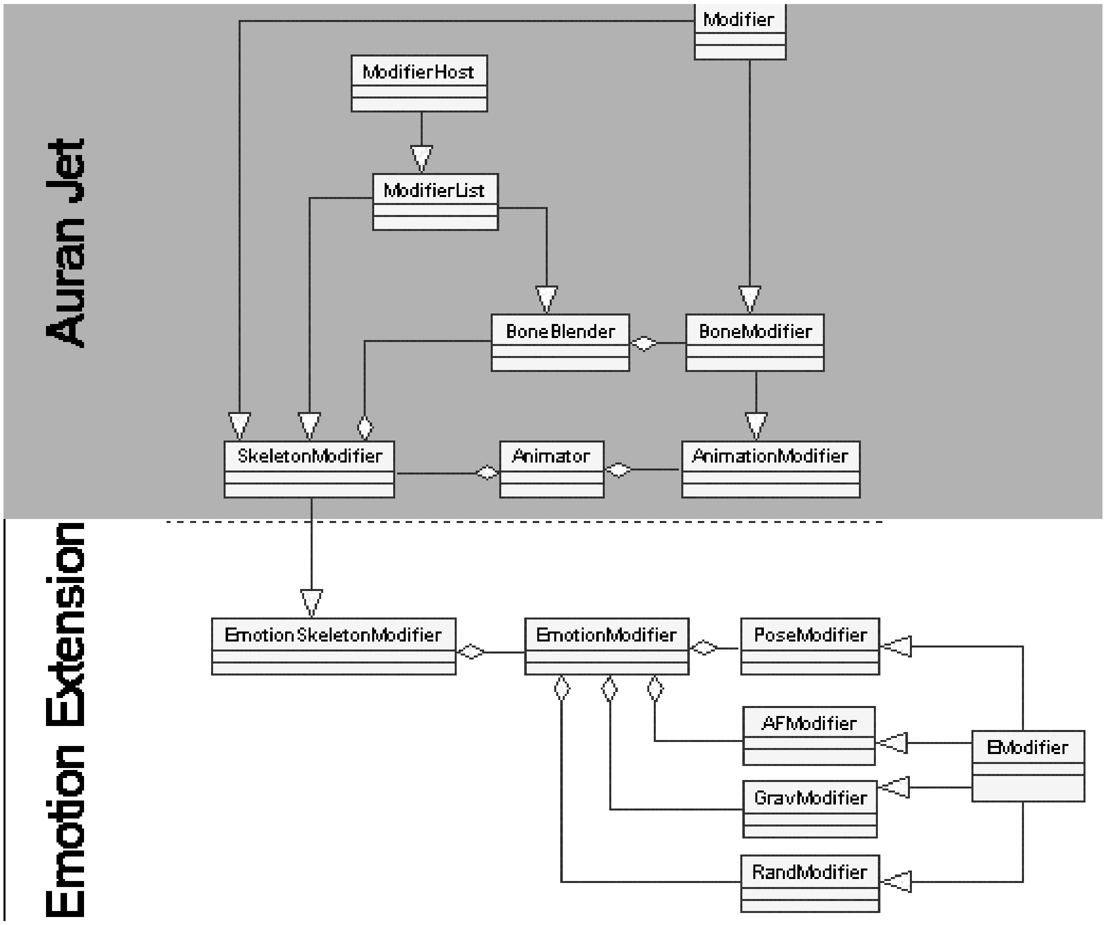


Illustration 11

How the emotion extension classes interact with Auran Jet

4.2 User Documentation

4.2.1 Emotion Demo

This program was initially used for evaluation of the effectiveness of emotions applied to a model. The main difference is that the emotion being applied is now displayed to the user. The controls for this program are quite simple. The keyboard numbers 1−5 loads an emotion while the letters underneath the numbers (Q,W,E,R,T,Y,U) load different animations. The emotion for number 1 is "none" so you are able to do a comparison before and after the emotion is applied. Pressing the ESC key exits the program.

This emotion demo defaults to 640x480 resolution within a window. However since the demo is itself a Auran Jet application so you can pass standard command line parameters to affect rendering mode/ screen resolution/windowing etc. (ie −1024 for 1024x768 resolution rendering. See the Auran Jet documentation for full details)

While this program allows you to smoothly blend between emotions, it is recommended that the user switch back to no emotion ("1" key) before changing the animation. If this is not done, the animation may in some circumstances look unusual. This program also shows some non−emotion effects using the modifiers and are bound to the keys 6−9.

For screen shots of this program see Appendix C.

4.2.2 Virtual Environment

The virtual environment was designed to demonstrate the modifiers working through the inference engine in an interactive world.

Commands for the virtual environment are:

Input	***Action***
Move mouse to screen edge	Camera scrolls in the direction of mouse movement
Click left mouse button	If the cursor is near a character, the character is selected. If the cursor is not near a character, any current selected character is unselected.
Up/Down Arrow key press	If no character is selected, the global influence of the current emotion is increased/decreased. If a character is selected, the influence this character has on the base animation is changed.
Left/Right Arrow key press	If no character is selected, the global emotion is changed. If a character is selected, the emotion specific to this character is changed.
Plus "+" key press	The camera zooms in.

Input	***Action***
Minus "−" key press	The camera zooms out.
"m" key press	Menu hidden/revealed
ESC key	Exist program.

The virtual environment demo defaults to fullscreen at 1024x768 but once again this is changeable using Auran Jet command line parameters.

It should be empasised that this is only a demo program and as such has some minor issues with it. The main issue involves collision detection as this has not been implemented properly and characters may get stuck or walk through objects. There is also a strange bug in Auran Jet where characters can sometimes disappear, but this is a rare occurrence and restarting the program usually resolves it. The inference engine also only has four pre−programmed emotions (calm, depressed, angry and happy) so extrapolated emotions may not look accurate.

For screen shots of this program see Appendix C

4.3 Resources

4.3.1 Programming Language/Libraries

All coding was done using Microsoft Visual C++ 6 and Beta 2 of Auran Jet.

4.3.2 Hardware/Software Required

In order to run the demo software the following is required:

- Virtual environment executable and emotion files
- Microsoft Windows 95/98/ME with Direct X
- Auran Jet runtime DLL’s (installed with program)
- Pentium II 400 or greater (Pentium III 600 recommended for the virtual environment)

- OpenGL based 3D accelerator (32MB GeForce recommended recommended for the virtual environment)
- 64MB RAM

4.4 Portability

As the foundation of this program is Auran Jet and no external libraries or tools are used, the implementation provided should be able to run on any system that supports Auran Jet. As of writing this only includes Microsoft Windows based operating systems. However, ports to Linux, PS2 and other platforms have been planned. (check www.auran.com)

4.5 Difficulties

The main difficulty encountered involved the fact that Auran Jet was still at beta stage at the time of programming. This meant that little documentation existed and much experimentation had to be carried out to gain access to the data structures. This also led to problems in implementing methods that required extensive use of mathematics as modifying some kinds of data proved difficult/impossible.

Another difficulty involved the fact that only the models provided by Auran Jet were available for testing of these methods. This means that it cannot be certain that these methods will work with most models in general.

Raw data on what different emotions look like was also unobtainable. The lead to the emotion pre−sets and rules in the virtual environment and inference engine being based on the creators limited experience and opinions. While this data was evaluated successfully, the system may not be accurate to a critical observer.

4.6 Limitations

While the overall aim of this project was to create a system where a user can simply alter the look of a model by specifying an emotion, some minor experimentation with different emotion values will probably be necessary when applying these emotions to a new object. This is due to the fact that a new model may be modified in unpredictable ways (ie hands through body) if limbs of the model are in strange positions.

Also, the programming of more emotions for the inference engine to use would be needed in a complete implementation.

5 Evaluation

Evaluation was carried out using both the Emotion Demo and virtual environment. An example evaluation sheet can bee found in Appendix B

5.1 Emotion Demo

The aim of the emotion demo was to discover how effective the modifiers were in portraying emotion and how well the input data for these modifiers was selected. The test participants were presented with a range animations applied to a human model. A emotion was then applied to each animation and the participant was asked to select from a list what emotion was they think was applied. Four emotions were tested on primarily three animations with users being able to select from a choice of over 20 emotions.

All participants responses were compared on the emotion wheel seen in illustration 3 and the number of degrees from the target emotion was calculated and averaged. The results for the main three animations are listed below.

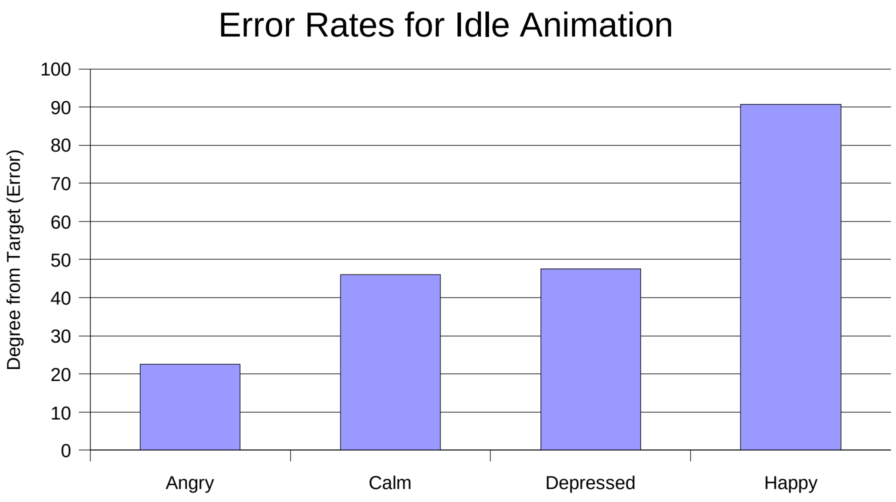


Illustration 12

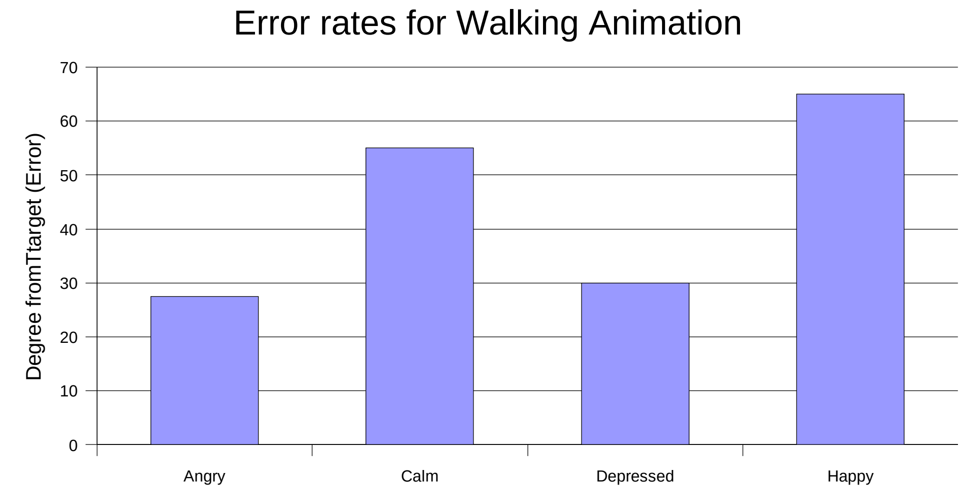


Illustration 13

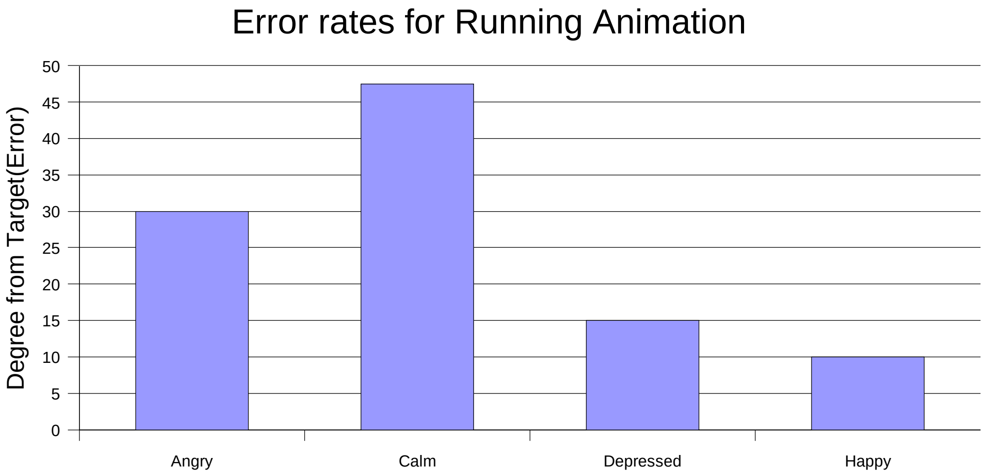


Illustration 14

The above results indicate that classification of emotions was generally acceptable (ie within 90^0 of target emotion) The following two tables take the data produced here to compare what emotions and animations caused the best classifications.

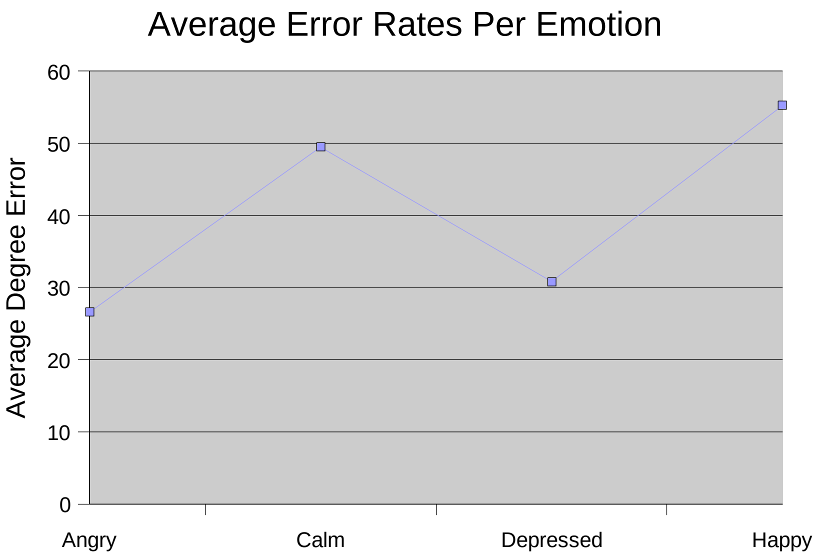


Illustration 15

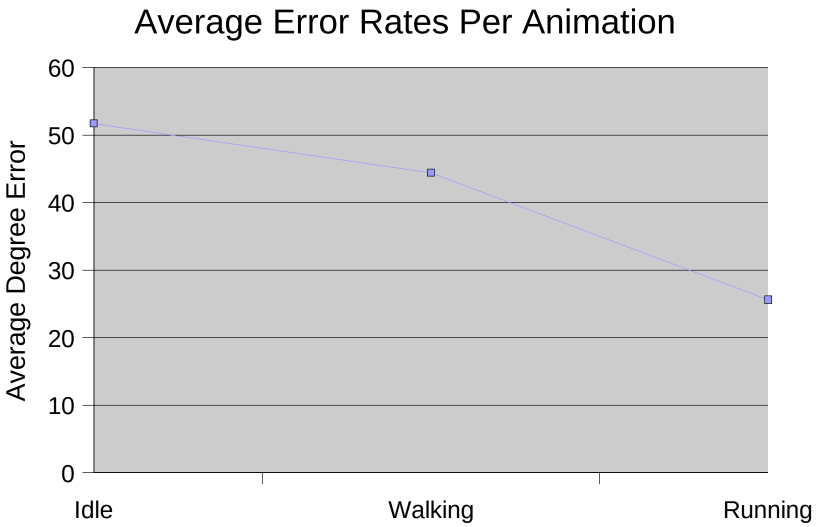


Illustration 16

The above tables seem to indicate that the angry and depressed emotions are best represented using the current modifiers. It also becomes apparent that animations that have more movement will produce higher levels of recognition in emotions.

5.2 Virtual Environment

The aim of evaluating the virtual environment was to discover if people were able to use the inference engine in a working application. Participants were asked to comment on the user interface and the general running of the system.

Most participants were able to use the interface after minimal practice but many requested additional features not relevant to the evaluation (such as online help, tooltips etc) Other requested features included a tracking system for selected characters so that they could easily see a character's emotion up close. The inference engine also proved to work generally as the users expected.

All participants were positive about the both programs, but due to the fact that only four people were available for testing, the results should not be classified as conclusive.

6 Extensions/Future Work

6.1 Immediate Extensions

Immediate extensions to this project would involve the refinement of the current modifiers. This would include reading ahead in the animations when calculating animation based data as well as optimising many of the modifiers for greater speed. (currently makes use of linked lists when array will do etc.) Specific refinement of the the gravity modifier could also be performed to implement the algorithm initially suggested for implementation .

Adding the ability to only apply emotions to specified bones and the ability of specifying which algorithm to use in each modifier, would also greatly enhance the usability of the modifiers in general projects.

In the inference engine, the loading of more pre−defined emotions would enhance the quality of the interpolated emotions produced

6.2 Short Term Extensions

In the short term, more research could be carried out into other modifiers and better algorithms for current modifiers. This is especially true for the random modifier as little research was done in comparison to the amount of data that exits on generating and applying randomness.

Other modifiers that could be applied include a type of limits modifier to ensure a character does not bend beyond certain limits and a perturbation modifier to alter the external mesh of the model.

In implementing a limits type modifier, it may be interesting to investigate the possibility of allowing the user to load a range of animations which are then analysed to determine what bone movements are valid for this model.

6.3 Long Term Future Work

In the long term as CPU power increases, more smoother and precise algorithms could be used instead of the linear interpolation currently implemented in the emotion modifiers. Also, the generation of a system that can store generalised emotion data for common objects (ie human) and apply this data to a new model would be useful in making the system rely less on user input. This would allow the animator to apply all the modifiers without loading in data such as neutral positions/orientations.

It also may be beneficial in the long term to research and implement a better inference engine for the system. This may involve simple pattern recognition techniques such as k−nearest neighbor all the way up to a multi−hidden layer neural network.

Incorporating into this research other areas such as facial expression should also be considered in the long term as the combined effects may produce very real results.

7 Conclusions

The aim of this research was to provide a means to aid animators or virtual environment designers apply emotions to their models. While a commercial level program specific to this task has not been developed in this research, much of the foundation work has been achieved to allow the creation of such a system.

Essential to the development of these types of emotion development systems would be the methods in which they modified the input models. The development of these types of algorithms has formed a core part of this research. Methods for altering the amplitude, frequency, pose and other attributes of a model have been proved successful and effective in representing emotion. While most research has been based on human/biped type of models, the design of these methods never relied on the type of models to achieve the emotional effects. (Although some modifiers require additional model data) It has also been shown that these methods are able to be implemented in real−time and therefore are able to be incorporated into virtual environments.

However, just having the ability to change a model’s appearance would detract from the goal of making the system available to non−technical people such as artists. This led to the development of a inference engine to accept aesthetic descriptors and infer emotional changes to a model. While the inference engine developed does rely on some pre−defined data and the resultant emotions are not always accurate, the general concept of taking scene emotions and combing them with individual emotions should prove useful in future work.

Example programs developed such as the Virtual Environment and Emotion demo prove that the implementation of these methods and concepts is possible. The limited evaluation of these programs has also shown that the emotions displayed can achieve impressive levels of accuracy.

As current technologies for computer animation and virtual environments continue to accelerate, applications and extensions of this research will undoubtedly start appearing in the not too distant future.

8 References

1. Amaya, K., *Emotion from Motion.* Proceeding of Graphics Interface, 1996. pg 96.
2. Rachel Price, C.D., Mervyn A. Jack, *An Investigation of the Effectiveness of Choreography for the Portrayal of Mood in Virtual Environments.* Agents ACM, 2000: p. 54−55.
3. Bates, J., *The Role of Emotion in Believable Agents.* Communications of the ACM, 1994. **37**(7): p. 122−125.
4. J.P. Lewis, M.C., Nickson Fong, *Pose Space Deformation: A Unified Approach to Shape Interpolation and Skeleton−Driven Deformation.* SIGGRAPH 2000, 2000: p. 165−172.
5. Maciejewski, M.G.a.A.A., *Computational Modeling for the Computer Animation of Legged Figures.* SIGGRAPH, 1985. **19**(3): p. 263−270.
6. Welman, C., *Inverse Kinematics and Geometric Constraints for Articulated Figure Manipulation*, in *Computing Science*. 1989, Simon Fraser University. p. 76.
7. Perlin, K., *Real Time Responsive Animation with Personality*. IEEE Transactions on Visualization and Computer Graphics, 1995. **1**(1): p. 5−15.
8. Jean−Sebastien Monzani, P.B., Ronan Boulic, Daniel Thalmann, *Using an Intermediate Skeleton and Inverse Kinematics for Motion Retargeting.* Computer Graphics Forum, 2000. **19**(3): p. 11−19.
9. Munetoshi Unuma, K.A., Royozo Takeuchi, *Fourier Principles for Emotion−based Human Figure Animation.* SIGGRAPH 1995, 1995: p. 91−96.
10. Jessica K Hodgins, N.S.P., *Adapting Simulated Behaviors for New Characters.* SIGGRAPH 1997, 1997: p. 153−162.
11. Lee, K., *Integration of Various Emotion Eliciting Factors for Life Like Actors.* IEEE SMC’99 Conference proceedings, 1999. **4**: p. 294−298.
12. Russell, J.A., *A Circumplex Model of Affect.* Journal of Personality and Social Psychology, 1980. **39**(6): p. 1161−1178.

13. Ken Perlin, A.G., *Improv: A System for Scripting Interactive Actors in Virtual Worlds.* SIGGRAPH 1996, 1996: p. 205−216.

14. Joseph W.Weiss, R. K. W. (1992). *5−Phase Project Management*, Perseus Books.

15. Leedy, P. D. (1974). *Practical Research: Planning and Design*, Macmillan Publishing Co. Inc.

16. Mary K. Ducharme, B. L. L., William A. Matthes,Rachel A. Vannatta (1997). *Comparison of Quantitative and Qualitative Research*. **2001**.[Web document]. Available: http://www.iptv.org/FINELINK/publications/criteria.html [2001, Feb 6].

17. Unknown (2000). *The Scientific Method.* **2001**.[Web document]. Available: http://va.essortment.com/scientificresea_rqce.htm [2001, Feb 6].

18. Wallace, U. (1995). *Deciding between Qualitative and Quantitative Research.* **2001**.[Web document]. Available: http://www.uwa.com/marketing/consultants/research.htm [2001, Feb 6].

19 *Yuencheng Lee, Demetri Terzopoulos, and Keith Waters.(1995) Realistic modeling for facial animation. In Computer Graphics (SIGGRAPH'87), pages 55−62, .*

20 Bruce Blumberg, (1999) *Swamped! Project description* **2001** [Web document] Available: http://characters.www.media.mit.edu/groups/characters/swamped/ [2001, June, 13]

Appendix A−Program Documentation

Emotion Modifier API

In order to add the Emotion Modifier classes to existing code there are only a few simple steps to follow:

1) If you current model code looks like this:

```
//Add a skeleton
skeleton = NewMem(SkeletonModifier(this, renderKernel));
//Add an animation interface
animator = NewMem(Animator(database, skeleton));
if(animator->Add(NULL,name,boneName)!=OK){
....
```

change the SkeletonModifier to a EmotionSkeletonModifier

```
//Add a skeleton
skeleton = NewMem(EmotionSkeletonModifier(this, renderKernel));
//Add an animation interface
animator = NewMem(Animator(database, skeleton));
if(animator->Add(NULL,name,boneName)!=OK){
....
```

2) In order to display an emotion, we have to load an EmotionModifier onto the EmotionSkeletonModifier

```
testEmotion=NewMem(EmotionModifier(skeleton));
```

3) Now load the modifiers you want onto the EmotionModifier. Note the paths for loading data such as pose data.

```
testEmotion->LoadModifier(MOD_POSE,1.00f,"Emotions","Happy","Human");
```

4) Your model now has the emotion applied, if you wish to modify individual modifier parameters you can access them through EmotionModifier−>GetModifier and then type cast.

NOTE: The file format for the pose and neutral data is very simple and a class called SkeletonFile can be used to create them.

Description Of Main Classes

EmotionSkeletonModifier – This is the main class which has access to the skeleton data by inheriting from SkeletonModifier. It’s only real purpose is to hold an array of EmotionModifiers.

EmotionModifier−This class holds a single emotion which may be made up of several Emodifiers. This class manages the loading and unloading of specific modifiers as well as skeleton manipulations.

PoseModifier− This class inherits from Emodifier and implements the pose methods discussed. You can only modify the influence value of this modifier once it is loaded.

AFModifier− This class inherits from Emodifier and implements the amplitude and frequency methods discussed. During run time you can alter the frequency, amplitude and influence values. The is an additional parameter where you can turn the frequency mixing on or off in case the animation speed gets modified elsewhere.

GravModifier−This class inherits from Emodifier and implements the gravity methods discussed. During run time you can alter the target gravity point and influence values. This modifier does not work that well and should be used sparingly

RandModifier− This class inherits from Emodifier and implements the random methods discussed. During run time you can alter the influence,variance, duration and percent random values. This class can produce very good results but should be used carefully as bones may intersect with each other.

Emodifier− This is the base class modifier. You cannot create instances of this class but if you want to creat your own modifiers you can inherit from this class.

VEInferenceEngine− This is a demo class implemeting the inference engine and has close ties with both VE characters and the EmotionModifiers. Requires modifiaction for general use.

Appendix B−Evaluation Document

PROGRAM EVALUATION QUESTIONAIRE

MODELING EMOTIONAL CHARACTERS IN VIRTUAL ENVIROMENTS

INFERENCING THE LOOK AND ANIMATION OF A CHARACTER BASED ON AESTHETIC DESCRIPTORS IN REAL TIME

Introduction

The two programs you will be evaluating are part of a Dissertation in an IT Honours course. The aim of this research is to alter the appearance of computer characters using aesthetic descriptors (ie pleased, glad, excited etc) in real time. These programs represent a implementation of developed experimental methods in order to achieve these goals.

Your evaluation will be used as a measure of how successful these methods work by how close you pick the target aesthetic descriptor when presented with a modified character.

The first part of the questionnaire simply asks some background information about yourself so that patterns in user responses may be obtained. This data will not be used for any other purpose.

The second part in the questionnaire involves displaying a single character on the screen and loading different emotions onto an array of character animations. As each emotion is presented you are asked to circle or write what you think the emotion is.

The final part of the questionnaire simply displays a virtual environment using these descriptors. You are asked if the user interface is easy to use and if changes that you make effect the virtual environment in the way you expect.

Please note that this is not a test on how well you can pick emotions as there there are no "correct answers".
This evaluation should take no more than 30 minutes.

QUESTIONNAIRE – Part 1

Background information:

1. Occupation: ___________________________________

2. Age: Under 18 Between 18 and 25 Between 25 and 40

   Between 40 and 60 Between 60 and 70 Over 70

3. Gender: Female Male

4. How would you rate your level of computer experience (Place a tick in only one box)

   Novice (rarely use a computer)

   Sometimes User (use a computer less than once a day)

   Home User

   Professional User (use a computer to complete your work)

   IT Professional

5. Indicate which of the following describes how often you use a virtual environment (ie computer games):

   Never Once a month Once a week Once a day

QUESTIONNAIRE – Part 2

In this part of the questionnaire you will need to run the "EmotionDemo.exe".

The controls for this program are quite simple. The numbers 1−5 loads an emotion while the letters underneath the numbers (Q,W,E,R,T,Y,U) load different animations. The emotion for number 1 is "none" so you are able to do a comparison before and after the emotion is applied. This is the default emotion upon starting the program.

Once you are in the program, the animation loaded initially is the idle animation.

1) Take a good look at it to get a feel for what the animation looks like.
2) Then press the "2" key to load emotion no2 and observe the changes.
3) Then circle in box below to indicate what emotion you think emotion 2 is. If the emotion is not listed, write it in the space provided.

Repeat the the above steps for emotions 3−5 and press 1 if you need to refresh you memory on what the animation looks like.

Idle Animation			
Emotion 2	***Emotion 3***	***Emotion 4***	***Emotion5***
Happy, Delighted, Excited, Astonished, Aroused, Tense, Alarmed, Angry, Afraid, Annoyed, Distressed, Frustrated, Miserable,Sad, Gloomy, Depressed, Bored, Droopy, Tired, Sleepy, Calm, Content, Glad **Other__________**	Happy, Delighted, Excited, Astonished, Aroused, Tense, Alarmed, Angry, Afraid, Annoyed, Distressed, Frustrated, Miserable,Sad, Gloomy, Depressed, Bored, Droopy, Tired, Sleepy, Calm, Content, Glad **Other__________**	Happy, Delighted, Excited, Astonished, Aroused, Tense, Alarmed, Angry, Afraid, Annoyed, Distressed, Frustrated, Miserable,Sad, Gloomy, Depressed, Bored, Droopy, Tired, Sleepy, Calm, Content, Glad **Other__________**	Happy, Delighted, Excited, Astonished, Aroused, Tense, Alarmed, Angry, Afraid, Annoyed, Distressed, Frustrated, Miserable,Sad, Gloomy, Depressed, Bored, Droopy, Tired, Sleepy, Calm, Content, Glad **Other__________**

Press "w" key for walk animation then "1" and repeat the above steps. Your answers may be different from the previous as this animation has much more movement to display the emotion.

Walk Animation			
Emotion 2	***Emotion 3***	***Emotion 4***	***Emotion5***
Happy, Delighted, Excited, Astonished, Aroused, Tense, Alarmed, Angry, Afraid, Annoyed, Distressed, Frustrated, Miserable,Sad, Gloomy, Depressed, Bored, Droopy, Tired, Sleepy, Calm, Content, Glad **Other__________**	Happy, Delighted, Excited, Astonished, Aroused, Tense, Alarmed, Angry, Afraid, Annoyed, Distressed, Frustrated, Miserable,Sad, Gloomy, Depressed, Bored, Droopy, Tired, Sleepy, Calm, Content, Glad **Other__________**	Happy, Delighted, Excited, Astonished, Aroused, Tense, Alarmed, Angry, Afraid, Annoyed, Distressed, Frustrated, Miserable,Sad, Gloomy, Depressed, Bored, Droopy, Tired, Sleepy, Calm, Content, Glad **Other__________**	Happy, Delighted, Excited, Astonished, Aroused, Tense, Alarmed, Angry, Afraid, Annoyed, Distressed, Frustrated, Miserable,Sad, Gloomy, Depressed, Bored, Droopy, Tired, Sleepy, Calm, Content, Glad **Other__________**

Once again repeat for the run animation by pressing "e" then "1"

Walk Animation			
Emotion 2	***Emotion 3***	***Emotion 4***	***Emotion5***
Happy, Delighted, Excited, Astonished, Aroused, Tense, Alarmed, Angry, Afraid, Annoyed, Distressed, Frustrated, Miserable,Sad, Gloomy, Depressed, Bored, Droopy, Tired, Sleepy, Calm, Content, Glad **Other__________**	Happy, Delighted, Excited, Astonished, Aroused, Tense, Alarmed, Angry, Afraid, Annoyed, Distressed, Frustrated, Miserable,Sad, Gloomy, Depressed, Bored, Droopy, Tired, Sleepy, Calm, Content, Glad **Other__________**	Happy, Delighted, Excited, Astonished, Aroused, Tense, Alarmed, Angry, Afraid, Annoyed, Distressed, Frustrated, Miserable,Sad, Gloomy, Depressed, Bored, Droopy, Tired, Sleepy, Calm, Content, Glad **Other__________**	Happy, Delighted, Excited, Astonished, Aroused, Tense, Alarmed, Angry, Afraid, Annoyed, Distressed, Frustrated, Miserable,Sad, Gloomy, Depressed, Bored, Droopy, Tired, Sleepy, Calm, Content, Glad **Other__________**

If you have any comments about the emotions, please write them below.

The letters R,T,Y,U also show animations from side walking to flapping. If you wish, you may repeat the above steps before writing your response below.

____

____

____

____

____

____

QUESTIONNAIRE – Part 3

In this part of the questionnaire you will need to run the "VE.exe". (Virtual Environment)

Following are the controls for the virtual environment:

Input	***Action***
Move mouse to screen edge	Camera scrolls in the direction of mouse movement
Click left mouse button	If the cursor is near a character, the character is selected. If the cursor is not near a character, any current selected character is unselected.
Up/Down Arrow key press	If no character is selected, the global influence of the current emotion is increased/decreased. If a character is selected, the influence this character has on the base animation is changed.
Left/Right Arrow key press	If no character is selected, the global emotion is changed. If a character is selected, the emotion specific to this character is changed.
Plus "+" key press	The camera zooms in.
Minus "−" key press	The camera zooms out.
"m" key press	Menu hidden/revealed

Once in the program, experiment with changing the global emotion and the emotions for specific characters.

Please Note:

- If the global influence is at 100%, all characters exhibit only the global emotion and not their specific emotion.
- If a character’s influence is a 0%, no emotions are applied to the character.
- The global emotion is combined with each character’s emotion (using the global influence) to produce the final emotion displayed. (ie set each the global influence to 0% for each character to have direct control over their emotion)

The following questions are about your usage of the virtual environment .

1) The amount of information which is contained on the screen can often affect the way that we perform a task. Did you find the screens to have an appropriate amount of information on them for your use?

   Too Little Appropriate Too Much

2) Did you find the general layout of the information on the screen to be appropriate for the tasks you were trying to perform?

   Yes No

   If you answered NO to this question please describe your reasons:

   ____

   ____

   ____

   ____

   ____

   ______

3) Did the system display the data in an easy to read format?

   Always Most of the time

   Some of the time Never

4) Did you find that you had any overall problems with the package?

   Yes No

   If you answered YES to this question please describe the difficulty you encountered: ___________

   ____

   ____

5) Whilst performing any set task did you find that you had some level of confusion as

to how to complete the task?

Yes No

If you answered YES to this question please indicate what the task was that you were trying to complete and at what stage you became confused

____

____

____

____

____

6) Were you aware of what the system was doing at all times (were there no unexplained pauses)?

Yes No

If you answered NO to this question please indicate where the unexplained pause occurred: _____

____

____

____

____

7) Indicate one way in which you think the product could be improved?________________________

____

____

____

____

8) How did you find the package's general ease of use?

Very difficult to use

Difficult to use

Easy to use

Very easy to use

If you found the package's ease of use difficult indicate why: ______________________________

____

____

____

9) Describe any aspects of this package that you found annoying or useless.

____

____

____

____

____

____

____

____

____

10)What suggestions would you make to improve the usability of the system?

____

____

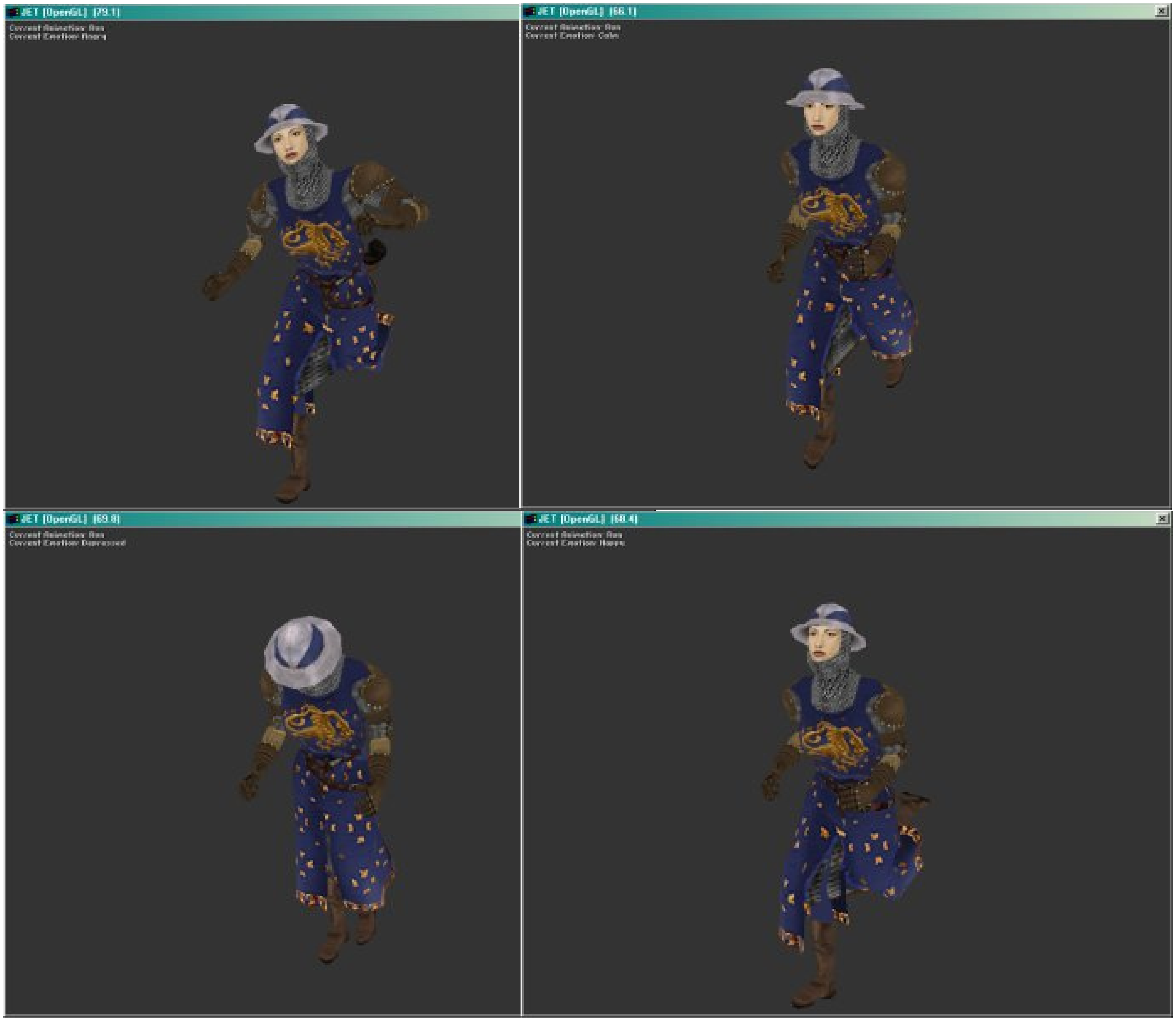
____

____

____

Appendix C−Screen Shots

The emotion demo screen shots of Anger,Calm,Depression and Happy (in that order) all using the running animation.



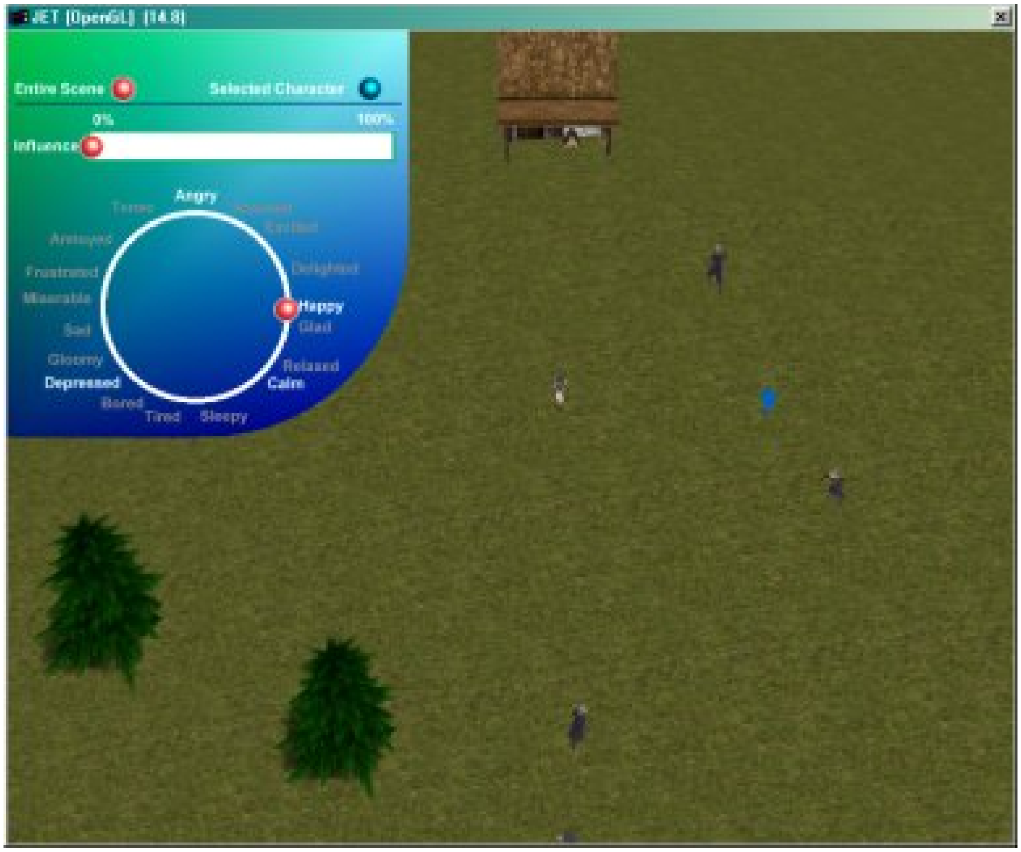
JET [OpenGL] (79.1)
Current Animation: Run
Current Emotion: Angry

JET [OpenGL] (66.1)
Current Animation: Run
Current Emotion: Calm

JET [OpenGL] (69.8)
Current Animation: Run
Current Emotion: Depressed

JET [OpenGL] (68.4)
Current Animation: Run
Current Emotion: Happy

A screen shot of the virtual environment demo


JET [OpenGL] (14.8)
Entire Scene
Selected Character
0%
100%
Influence
Angry
Tense
Astonished
Excited
Annoyed
Delighted
Frustrated
Miserable
Happy
Glad
Sad
Gloomy
Relaxed
Depressed
Calm
Bored
Tired
Sleepy